8. Prashant

Program Name: Prashant.java **Input File: prashant.dat**

Prashant has just learned about permutations and knows how to calculate how many possible unique permutations can be generated from a simple list of items. For example, he understands that given three elements A, B and C, there are six different ways these three elements can be listed in order: ABC, ACB, BAC, BCA, CAB, and CBA. A way to express the total is to use the factorial function in mathematics, expressed as 3!, 3 being the number of elements, and the ! operator indicating a "count down and multiply" process, like this:

3 x 2 x 1 = 6

Furthermore he learned that if there are duplicates, like in the set of elements A, B, C and C, there are indeed 4!, or 24 permutations, but since C is duplicated, there are only half as many unique permutations, which would be 12:

ABCC, ACBC, ACCB, BACC, BCAC, BCCA, CABC, CACB, CBAC, CBCA, CCAB, CCBA

The general expression to reduce the value due to duplicates would be **num!/dup!**, where **num** is the number of elements, and **dup** is the number of duplicates. This formula would still work if the value of **dup** was zero (indicating no duplicates), resulting in 0!, or 1, since by definition, 0! is equal to 1. For the two examples so far, the calculations would be:

3!/0! = 6
4!/2! = 12

However, things got a bit more interesting when he tried the sequence A B A B, which had **two** elements duplicated. He discovered that he needed to alter the denominator of the formula, accounting for the multiple duplicates, resulting in this formula, **num!/(dup1!* dup2!),** which would produce the expression:

4!/(2!*2!) = 6

With even more complex sequences, for each element that was duplicated, he had to add another factor to the denominator to account for each additional duplication. For instance, the sequence A B C A B C D A has three elements that are duplicated, one even occurring three times, requiring a denominator of 3!*2!*2! The resulting number of unique permutations for this list is 630.

Prashant realized this was getting quite complicated and decided to reach out to you for help. He is only interested in calculating the number of **unique permutations** that would be generated, given a list of elements, and not the actual permutations. Can you help him write a program to do this?

Input: Several lines of data, each containing a list of elements represented by single uppercase letters, with single space separation, each sequence possibly containing duplicate elements as demonstrated above. There will be no more than 15 elements in each line of data.

Output: The total number of unique permutations that would be generated with the given data, eliminating any duplicate permutations due to duplicate elements.

Sample input:

```
A B C
A B C C
A B A B
A B C A B C D
```

Sample output:

```
6
12
6
630
```